

Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

**Presented by MANICKAM RAVISEKAR ,
Master of Science in Computer Science,
19599 , Spring 2022**

**Guidance from Dr., Professor Henry Chang
TA : Zizhuo Huang**

**SAN FRANCISCO BAY UNIVERSITY
47671 WestingHouse Dr., Fremont, CA 94539**

ACKNOWLEDGEMENT

One of our Master's Degree Project for

Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE HTTP JSON using JavaScript for the time server was an Interesting, made me to learn new things, it is useful in designing and applying on Google Cloud – Kubernetes.

For deploying this project , I would like to thank Dr. Henry Chang and Zizhuo Huang.

Also, for all I would like to always pray to Almighty for giving us wisdom and power to understand things.

Conclusion

Application usage of student server and bookshelf:

- 1) Student Server program is use to get students details
- 2) By REST API using key field of student id
- 3) Book Shelf Application allows to manage books by adding new books, modifying books and deleting books .

Step1 Create MongoDB using Persistent Volume on GKE, and insert records into it :-

1. Create a cluster as usual on GKE

gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1

The screenshot shows the Google Cloud Platform console interface. At the top, there's a navigation bar with 'Google Cloud Platform' and 'My Project 31985'. Below this, there are tabs for 'DASHBOARD', 'ACTIVITY', and 'RECOMMENDATIONS'. The main content area displays three widgets: 'Project info' (showing 'My Project 31985'), 'API APIs' (showing 'Requests (requests/sec)' at 1.0), and 'Google Cloud Platform status' (showing 'All services normal'). Below these widgets, there's a 'CLOUD SHELL Terminal' window. The terminal output shows the command to create a Kubernetes cluster: `gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1`. The output indicates that the cluster 'kubia' is being created in the 'us-west1' region with 1 node of type 'e2-micro'. The terminal also shows the cluster's status as 'RUNNING' and provides the URL to inspect the cluster: `https://console.cloud.google.com/kubernetes/workload/_gcloud/us-west1/kubia?project=key-tokenizer-344403`.

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to key-tokenizer-344403.
Use "gcloud config set project (PROJECT_ID)" to change to a different project.
manickam19599@cloudshell:~ (key-tokenizer-344403)$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, please pass the '--no-enable-ip-alias' flag
Note: Your Pod address range ('--cluster-ipv4-cidr') can accommodate at most 1008 node(s).
Creating cluster kubia in us-west1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/key-tokenizer-344403/zones/us-west1/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-west1/kubia?project=key-tokenizer-344403
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 35.247.28.248
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 3
STATUS: RUNNING
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

Let's create a Persistent Volume first,
gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb

The screenshot shows the Google Cloud Platform console interface. At the top, there's a navigation bar with the Google Cloud Platform logo, the project name 'My Project 31985', and a search bar. Below this, there are tabs for 'DASHBOARD', 'ACTIVITY', and 'RECOMMENDATIONS'. The main content area displays three widgets: 'Project info' (showing project name 'My Project 31985' and project number), 'API APIs' (showing requests per second), and 'Google Cloud Platform status' (showing 'All services normal').

Below the widgets, there's a 'CLOUD SHELL' section with a terminal window. The terminal shows the following commands and output:

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https://developers.google.com/compute/docs/disks#performance.
ERROR: (gcloud.compute.disks.create) Could not fetch resource:
- The resource 'projects/key-tokenizer-344403/zones/us-west1-a/disks/mongodb' already exists

manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

The terminal window has a title bar that says 'Terminal (key-tokenizer-344403)' and an 'Open Editor' button. The bottom of the screenshot shows the Windows taskbar with various application icons and the system clock indicating 6:43 PM on 3/17/2022.

kubectl apply -f mongodb-deployment.yaml: **Wait until pod starts running**

The screenshot shows a web browser window displaying the Google Cloud Platform console. The address bar shows the URL: `console.cloud.google.com/home/dashboard?project=key-tokenizer-344403&organizationId=831774947012&cloudshell=true`. The page header includes the Google Cloud Platform logo, the project name "My Project 31985", and a search bar. The main content area is a terminal window titled "Terminal" for the project "key-tokenizer-344403". The terminal shows the following commands and output:

```
manickam19599@cloudshell:~ (key-tokenizer-344403) $ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
manickam19599@cloudshell:~ (key-tokenizer-344403) $
```

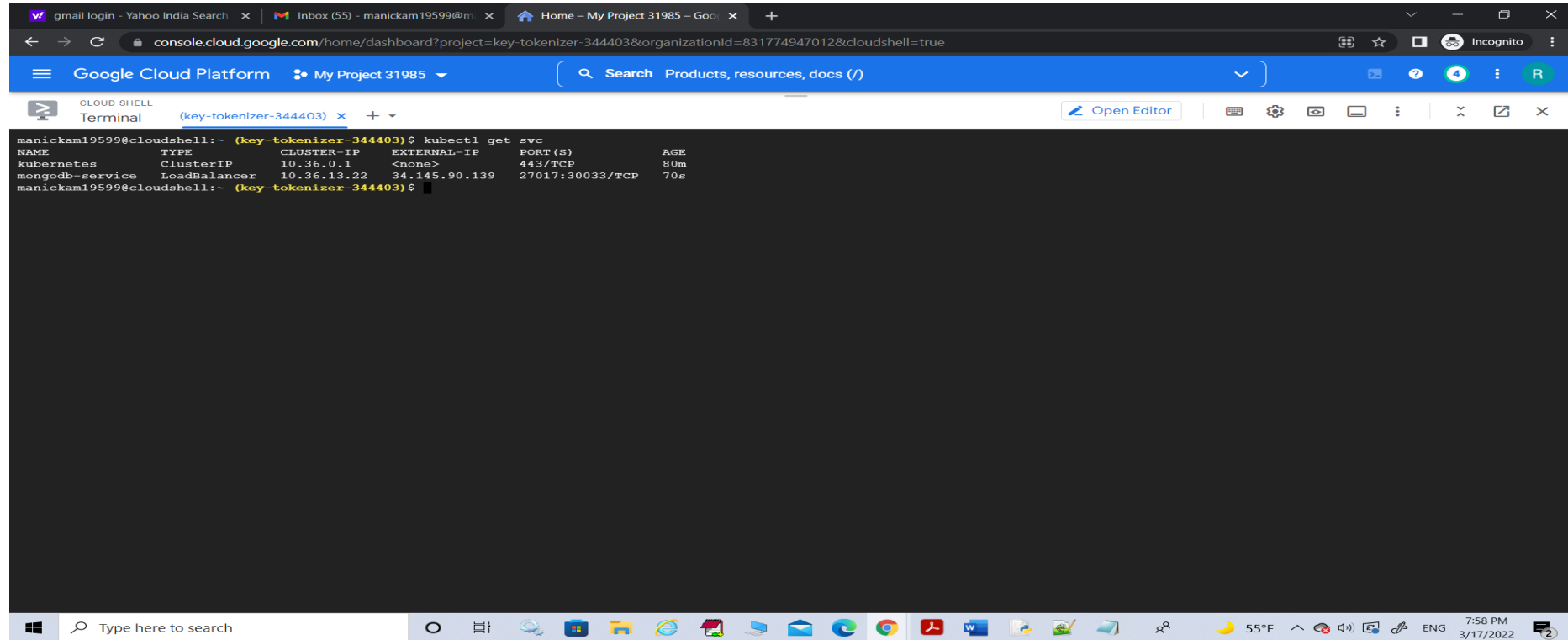
The Windows taskbar is visible at the bottom of the screen, showing the Start button, a search bar, and several application icons including File Explorer, Edge, and various utility programs. The system clock in the bottom right corner indicates the time is 7:19 PM on 3/17/2022.

```
kubectl apply -f mongodb-service.yaml
```

The screenshot shows a web browser window with the Google Cloud Platform console. The address bar displays the URL: `console.cloud.google.com/home/dashboard?project=key-tokenizer-344403&organizationId=631774947012&cloudshell=true`. The page header includes the Google Cloud Platform logo, the project name "My Project 31985", a search bar, and links to "Products, resources, docs". Below the header, a "CLOUD SHELL Terminal" tab is active, showing a terminal session. The terminal prompt is `manickam19599@cloudshell:~`. The user has entered the command `kubectl create -f mongodb-service.yaml`, and the output is `service/mongodb-service created`. The terminal window has a toolbar with an "Open Editor" button and various icons for file management and settings. The bottom of the image shows a Windows taskbar with a search bar and several application icons.

6. Wait couple of minutes, and check if the service is up

kubectl get svc Please wait until you see the external-ip is generated for mongodb-service, then you can move forward



The screenshot shows the Google Cloud Platform console in a web browser. The terminal window displays the output of the command `kubectl get svc`. The output is a table with columns: NAME, TYPE, CLUSTER-IP, EXTERNAL-IP, PORT(S), and AGE. The table lists two services: `kubernetes` and `mongodb-service`. The `mongodb-service` has an external IP of `34.145.90.139`.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.36.0.1	<none>	443/TCP	80m
mongodb-service	LoadBalancer	10.36.13.22	34.145.90.139	27017:30033/TCP	70s

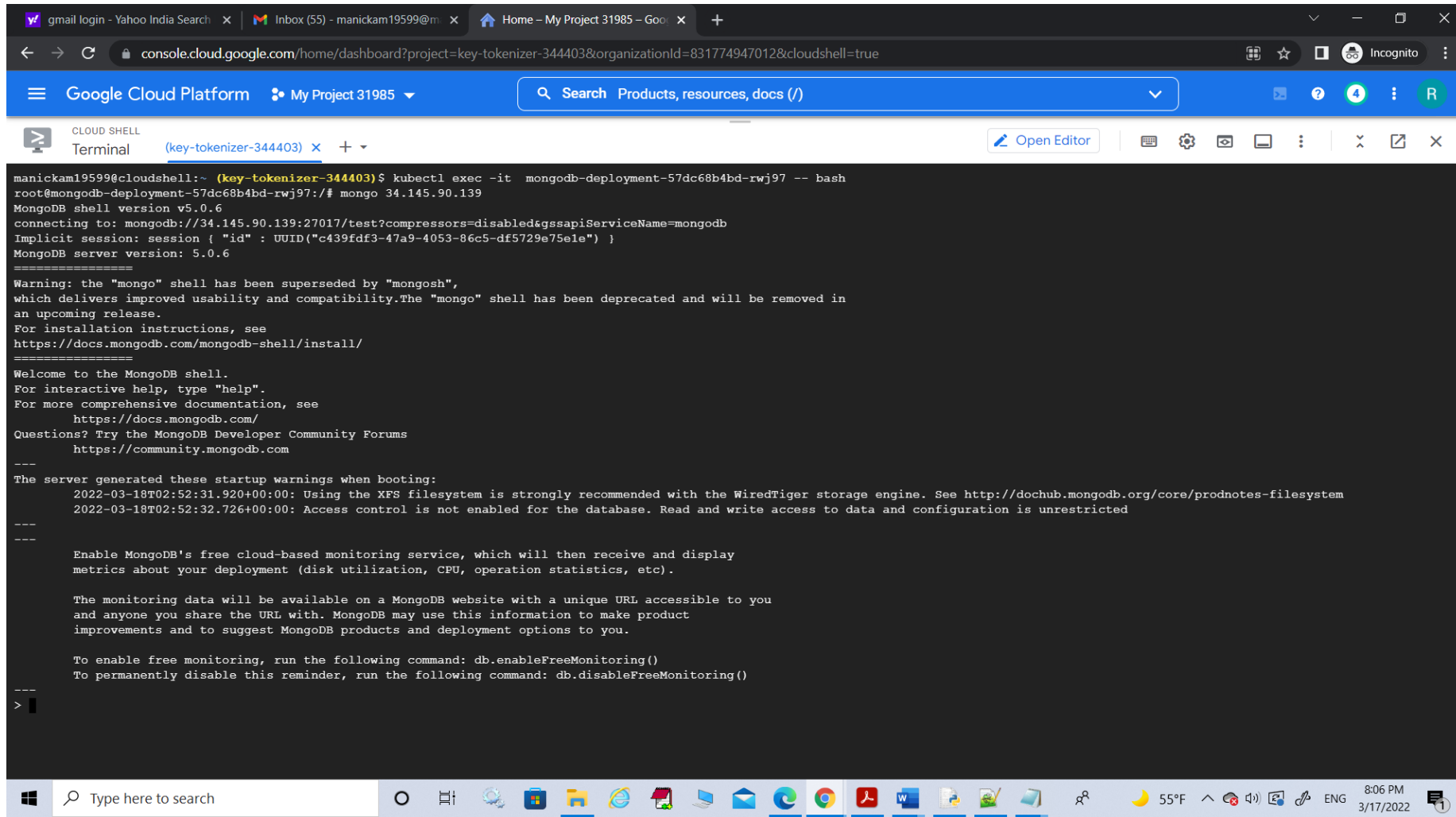
Now try and see if mongoDB is functioning for connections using the External-IP

kubectl exec -it mongodb-deployment-**replace-with-your-pod-name** -- bash

Now you are inside the mongodb deployment pod

Try : mongo External-IP

You should see something like this, which means your mongoDB is up and can be accessed using the External-IP : mongo 34.145.90.139



```
manickam19599@cloudshell:~ (key-tokenizer-344403) $ kubectl exec -it mongodb-deployment-57dc68b4bd-rwj97 -- bash
root@mongodb-deployment-57dc68b4bd-rwj97:/# mongo 34.145.90.139
MongoDB shell version v5.0.6
connecting to: mongodb://34.145.90.139:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c439fdf3-47a9-4053-86c5-df5729e75e1e") }
MongoDB server version: 5.0.6
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
https://community.mongodb.com
---
The server generated these startup warnings when booting:
2022-03-18T02:52:31.920+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-03-18T02:52:32.726+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```


.exit

The screenshot shows a web browser window with the Google Cloud Platform console. The address bar shows the URL: `console.cloud.google.com/home/dashboard?project=key-tokenizer-344403&organizationId=831774947012&cloudshell=true`. The page header includes the Google Cloud Platform logo, the project name "My Project 31985", a search bar, and a user profile icon. Below the header, there is a "CLOUD SHELL" tab labeled "Terminal (key-tokenizer-344403)". The terminal window shows a session where the user runs `kubectl exec -it mongodb-deployment-57dc68b4bd-rwj97 -- bash` to access a MongoDB pod. Inside the pod, the user runs `mongo`, which starts the MongoDB shell. The shell displays version information and a warning about the "mongo" shell being superseded by "mongosh". The user then types `> exit`, which returns them to the cloudshell prompt. The terminal output is as follows:

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl exec -it mongodb-deployment-57dc68b4bd-rwj97 -- bash
root@mongodb-deployment-57dc68b4bd-rwj97:/# mongo 34.145.90.139
MongoDB shell version v5.0.6
connecting to: mongodb://34.145.90.139:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c439fdE3-47a9-4053-86c5-df5729e75e1e") }
MongoDB server version: 5.0.6
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
https://community.mongodb.com
---
The server generated these startup warnings when booting:
  2022-03-18T02:52:31.920+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
  2022-03-18T02:52:32.726+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> exit
bye
root@mongodb-deployment-57dc68b4bd-rwj97:/# exit
exit
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

The bottom of the image shows the Windows taskbar with the search bar, task view button, and several application icons. The system tray on the right shows the date and time as 8:07 PM on 3/17/2022, along with weather and network status icons.

```

Enter the following line by line
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://34.82.40.170/mydb"
// Connect to the db
MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true },
function(err, client){
  if (err)
    throw err;
  // create a document to be inserted
  var db = client.db("studentdb");
  const docs = [
    { student_id: 11111, student_name: "Bruce Lee", grade: 84},
    { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
    { student_id: 33333, student_name: "Jet Li", grade: 88}
  ]
  db.collection("students").insertMany(docs, function(err, res){
    if(err) throw err;
    console.log(res.insertedCount);
    client.close();
  });
  db.collection("students").findOne({"student_id": 11111},
  function(err, result){
    console.log(result);
  });
});
});

```

**We need to insert some records into the mongoDB for later use
node use : node and do the entry as given beside and check you
can see 3 (records created)**

The screenshot shows the Google Cloud Platform console with a terminal window open. The terminal displays the execution of a Node.js script that connects to a MongoDB database and inserts three records. The output shows the script running successfully and returning 3 records inserted.

```

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
manickam19599@cloudshell:~ (key-tokenizer-344403)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://35.230.95.35/mydb"
undefined
> // Connect to the db
undefined
> MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true },
... function(err, client){
...   if (err)
...     throw err;
...   // create a document to be inserted
...   var db = client.db("studentdb");
...   const docs = [
...     { student_id: 11111, student_name: "Bruce Lee", grade: 84},
...     { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
...     { student_id: 33333, student_name: "Jet Li", grade: 88}
...   ]
...   db.collection("students").insertMany(docs, function(err, res){
...     if (err) throw err;
...     console.log(res.insertedCount);
...     client.close();
...   });
...   db.collection("students").findOne({"student_id": 11111},
...   function(err, result){
...     console.log(result);
...   });
... })
undefined
> null

```

Modify our studentServer to get records from MongoDB and deploy to GKE

:- studentServer.js page 1 / 2

```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
  MONGO_URL,
  MONGO_DATABASE
} = process.env;
var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to the db
console.log(uri);
var server = http.createServer(function(req, res) {
  var result;
  // req.url = /api/score?student_id=11111
  var parsedUrl = url.parse(req.url, true);
  var student_id = parseInt(parsedUrl.query.student_id);
```

```
// match req.url with the string /api/score
if (/^\/api\/score/.test(req.url)) {
  // e.g., of student_id 1111
  MongoClient.connect(uri, {
    useNewUrlParser: true,
    useUnifiedTopology: true
  }, function(err, client) {
    if (err)
      throw err;
    var db = client.db("studentdb");
    db.collection("students").findOne({
      "student_id": student_id
    },
    (err, student) => {
      if (err)
        throw new Error(err.message, null);
      if (student) {
        res.writeHead(200, {
          'Content-Type': 'application/json'
        })
        res.end(JSON.stringify(student) + '\n')
      } else {
        res.writeHead(404);
        res.end("Student Not Found \n");
      }
    });
  });
  res.writeHead(404);
  res.end("Wrong url, please try again\n");
}
});
server.listen(8080);
```

Modify our studentServer to get records from MongoDB and deploy to GKE :- studentServer.js page 2 / 2

Create Dockerfile

```
FROM node:7
```

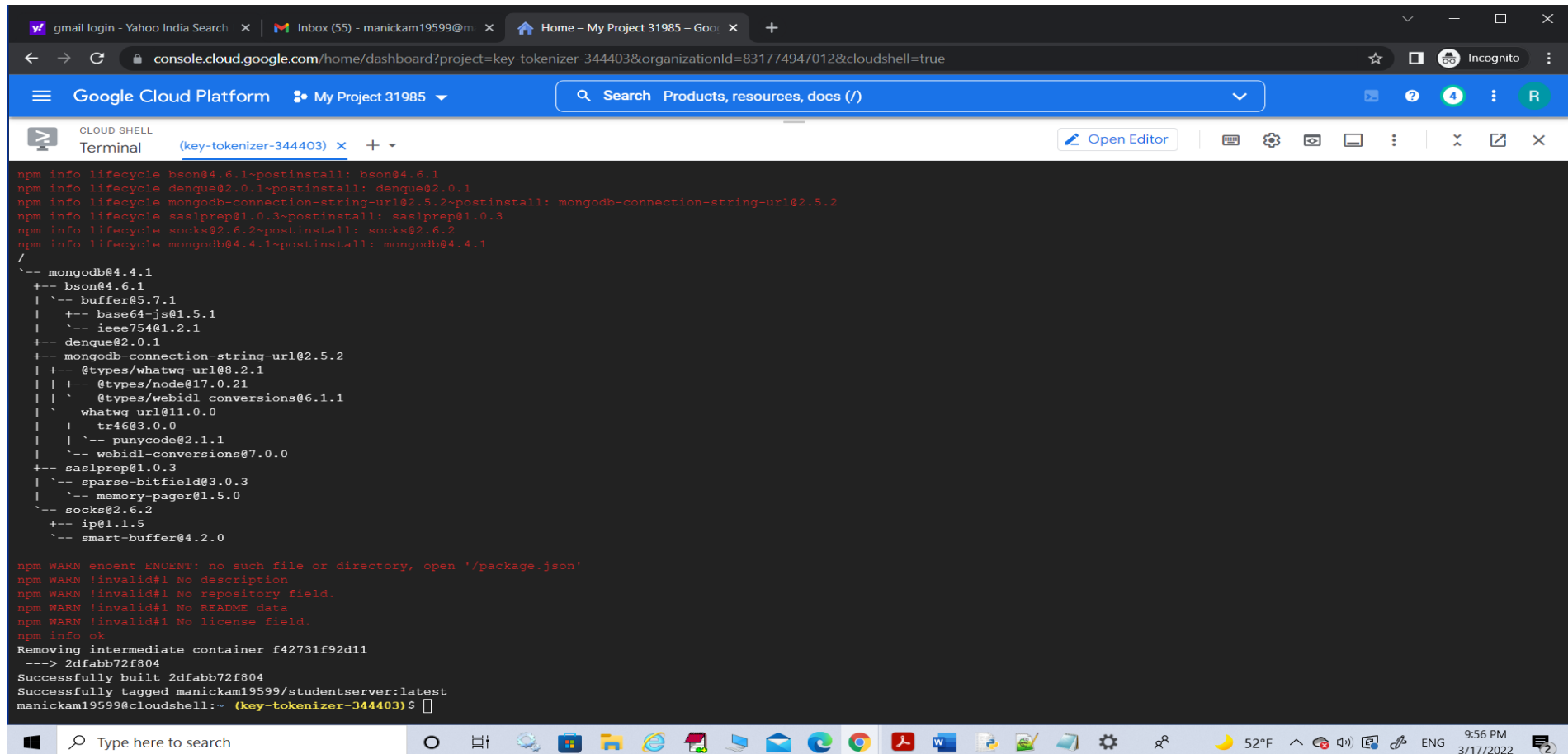
```
ADD studentServer.js
```

```
/studentServer.js
```

```
ENTRYPOINT ["node",  
"studentServer.js"]
```

```
RUN npm install mongodb
```

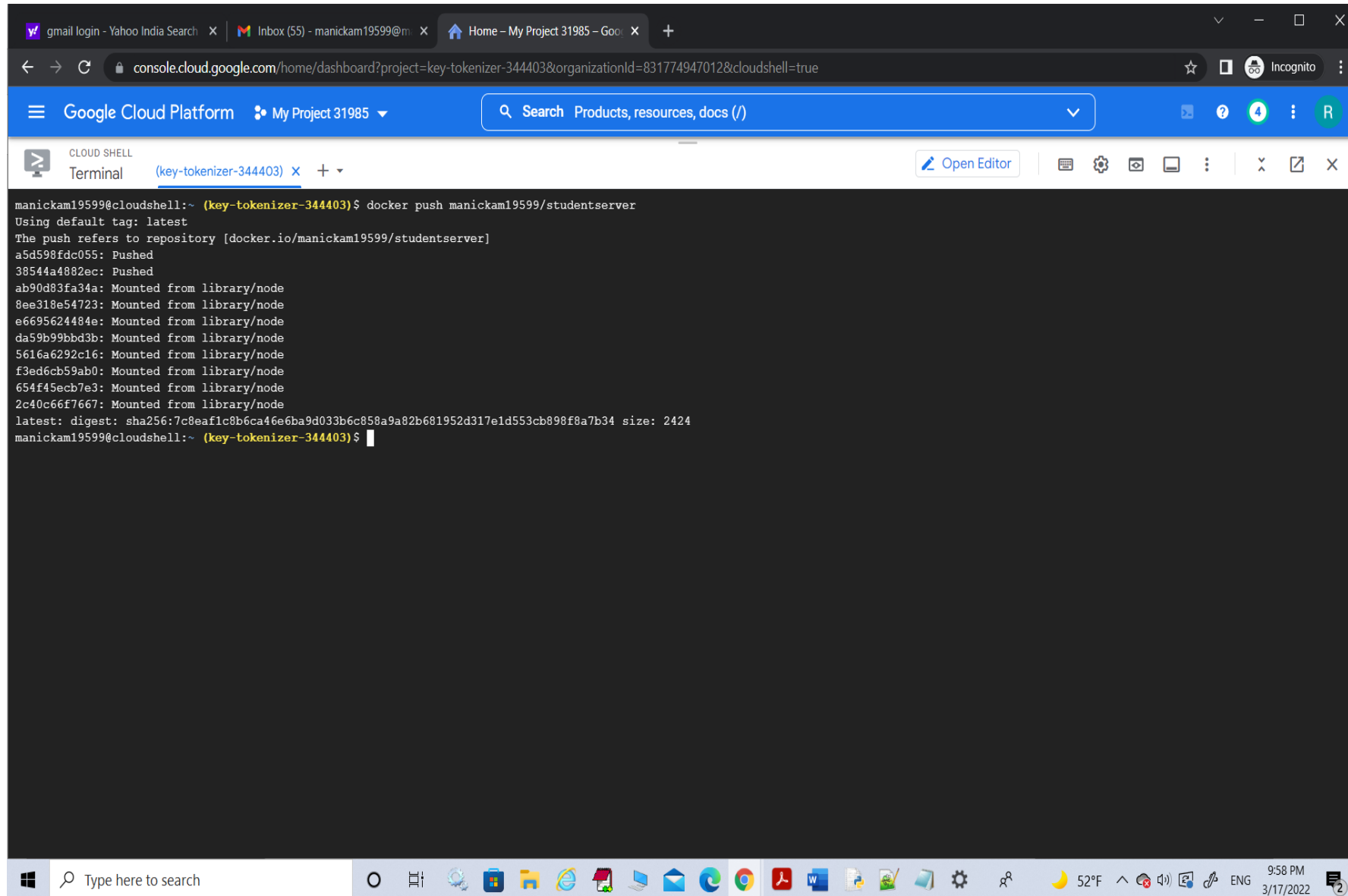
docker build -t < yourdockerhubID >/studentserver ensure no errors ignore versions warning



The screenshot shows a Google Cloud Platform Cloud Shell terminal window. The terminal displays the output of a Docker build command. It starts with npm lifecycle messages for various dependencies: bson@4.6.1, denque@2.0.1, mongodb-connection-string-url@2.5.2, saslprep@1.0.3, socks@2.6.2, and mongodb@4.4.1. The build process then lists the installed packages and their versions, including bson@4.6.1, buffer@5.7.1, base64-js@1.5.1, ieee754@1.2.1, denque@2.0.1, mongodb-connection-string-url@2.5.2, @types/whatwg-url@8.2.1, @types/node@17.0.21, @types/webidl-conversions@6.1.1, whatwg-url@11.0.0, tr46@3.0.0, punycode@2.1.1, webidl-conversions@7.0.0, saslprep@1.0.3, sparse-bitfield@3.0.3, memory-pager@1.5.0, socks@2.6.2, ip@1.1.5, and smart-buffer@4.2.0. The build is successful, and the container is tagged as manickam19599/studentserver:latest. The terminal also shows some npm warnings about missing fields in the package.json file.

```
npm info lifecycle bson@4.6.1~postinstall: bson@4.6.1
npm info lifecycle denque@2.0.1~postinstall: denque@2.0.1
npm info lifecycle mongodb-connection-string-url@2.5.2~postinstall: mongodb-connection-string-url@2.5.2
npm info lifecycle saslprep@1.0.3~postinstall: saslprep@1.0.3
npm info lifecycle socks@2.6.2~postinstall: socks@2.6.2
npm info lifecycle mongodb@4.4.1~postinstall: mongodb@4.4.1
/
-- mongodb@4.4.1
+-- bson@4.6.1
|   |-- buffer@5.7.1
|   |   |-- base64-js@1.5.1
|   |   |-- ieee754@1.2.1
+-- denque@2.0.1
+-- mongodb-connection-string-url@2.5.2
|   |-- @types/whatwg-url@8.2.1
|   |   |-- @types/node@17.0.21
|   |   |-- @types/webidl-conversions@6.1.1
|   |   |-- whatwg-url@11.0.0
|   |   |-- tr46@3.0.0
|   |   |   |-- punycode@2.1.1
|   |   |   |-- webidl-conversions@7.0.0
+-- saslprep@1.0.3
|   |-- sparse-bitfield@3.0.3
|   |-- memory-pager@1.5.0
+-- socks@2.6.2
|   |-- ip@1.1.5
|   |-- smart-buffer@4.2.0
npm WARN enoent ENOENT: no such file or directory, open '/package.json'
npm WARN invalid#1 No description
npm WARN invalid#1 No repository field.
npm WARN invalid#1 No README data
npm WARN invalid#1 No license field.
npm info ok
Removing intermediate container f42731f92d11
--> 2dfabb72f804
Successfully built 2dfabb72f804
Successfully tagged manickam19599/studentserver:latest
manickam19599@cloudshell:~ (key-tokenizer-344403) $
```

Push the docker image
docker push yourdockerhubID/studentserver



The screenshot shows a web browser window with the Google Cloud Platform console. The browser's address bar displays the URL: `console.cloud.google.com/home/dashboard?project=key-tokenizer-344403&organizationId=831774947012&cloudshell=true`. The console header includes the Google Cloud Platform logo, the project name "My Project 31985", and a search bar. Below the header, a "CLOUD SHELL" terminal window is open for the project "key-tokenizer-344403". The terminal shows the execution of the command `docker push manickam19599/studentserver`. The output indicates that the image was pushed successfully, showing the repository path `docker.io/manickam19599/studentserver` and a list of layers being pushed, including `latest` and several intermediate layers. The final output shows the digest and size of the `latest` tag: `latest: digest: sha256:7c8eaf1c8b6ca46e6ba9d033b6c858a9a82b681952d317e1d553cb898f8a7b34 size: 2424`. The terminal prompt is `manickam19599@cloudshell:~ (key-tokenizer-344403)$`. The Windows taskbar is visible at the bottom of the screen, showing the search bar and various application icons.

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ docker push manickam19599/studentserver
Using default tag: latest
The push refers to repository [docker.io/manickam19599/studentserver]
a5d598fdc055: Pushed
38544a4882ec: Pushed
ab90d83fa34a: Mounted from library/node
8ee318e54723: Mounted from library/node
e6695624484e: Mounted from library/node
da59b99bbd3b: Mounted from library/node
5616a6292c16: Mounted from library/node
f3ed6cb59ab0: Mounted from library/node
654f45ecb7e3: Mounted from library/node
2c40c66f7667: Mounted from library/node
latest: digest: sha256:7c8eaf1c8b6ca46e6ba9d033b6c858a9a82b681952d317e1d553cb898f8a7b34 size: 2424
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

```

from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os
app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://" + os.getenv("MONGO_URL") + "/" + os.getenv("MONGO_DATABASE")
app.config['JSONIFY_PRETTYPRINT_REGULAR'] = True
mongo = PyMongo(app)
db = mongo.db
@app.route("/")
def index():
    hostname = socket.gethostname()
    return jsonify(
        message="Welcome to bookshelf app! I am running inside {} pod!".format(hostname)
    )
@app.route("/books")
def get_all_tasks():
    books = db.bookshelf.find()
    data = []
    for book in books:
        data.append({
            "id": str(book["_id"]),
            "Book Name": book["book_name"],
            "Book Author": book["book_author"],
            "ISBN" : book["ISBN"]
        })
    return jsonify(
        data
    )

```

Create a python Flask bookshelf REST API and deploy on GKE :- bookshelf.py - page 1 / 4

Create a python Flask bookshelf REST API and deploy on GKE :- page bookshelf.py - 2 / 4

```
@app.route("/books")
def get_all_tasks():
    books = db.bookshelf.find()
    data = []
    for book in books:
        data.append({
            "id": str(book["_id"]),
            "Book Name": book["book_name"],
            "Book Author": book["book_author"],
            "ISBN" : book["ISBN"]
        })
    return jsonify(
        data
    )

@app.route("/book", methods=["POST"])
def add_book():
    book = request.get_json(force=True)
    db.bookshelf.insert_one({
        "book_name": book["book_name"],
        "book_author": book["book_author"],
        "ISBN": book["isbn"]
    })
    return jsonify(
        message="Task saved successfully!"
    )
```


Create a python Flask bookshelf REST API and deploy on GKE :-

Page bookshelf.py - 3 / 4

```
@app.route("/book/<id>", methods=["PUT"])
def update_book(id):
    data = request.get_json(force=True)
    print(data)
    response = db.bookshelf.update_many ({ "_id": ObjectId(id)},
        {"$set": {"book_name": data["book_name"], "book_author": data["book_author"], "ISBN":
data["isbn"]}}
    )
    if response.matched_count:
        message = "Task updated successfully!"
    else:
        message = "No book found!"
    return jsonify(
        message=message
    )

@app.route("/book/<id>", methods=["DELETE"])
def delete_task(id):
    response = db.bookshelf.delete_one({"_id": ObjectId(id)})
    if response.deleted_count:
        message = "Task deleted successfully!"
    else:
        message = "No book found!"
    return jsonify(
        message=message
    )
```

Create a python Flask bookshelf REST API and deploy on GKE

page bookshelf.py - 3 / 4

```
@app.route("/tasks/delete",
methods=["POST"])
def delete_all_tasks():
    db.bookshelf.remove()
    return jsonify(
        message="All Books deleted!"
    )

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Create a Dockerfile

```
FROM python:alpine3.7
```

```
COPY . /app
```

```
WORKDIR /app
```

```
RUN pip install -r requirements.txt
```

```
ENV PORT 5000
```

```
EXPOSE 5000
```

```
ENTRYPOINT [ "python3" ]
```

```
CMD [ "bookshelf.py" ]
```

Contents of file requirements.txt

Flask

Flask-PyMongo

Build the bookshelf app into a docker image `docker build -t < yourdockerhubID >/bookshelf`. Make sure this step build successfully

The screenshot shows a terminal window in Google Cloud Platform. The user is in a shell environment with the prompt `manickam19599@cloudshell:~ (key-tokenizer-344403)`. They first run `docker images` to list existing images, showing `python:alpine3.7` and `node:7`. Then they run `docker build -t manickam19599/bookshelf .`. The build process is shown in steps: Step 1/8 (FROM), Step 2/8 (COPY), Step 3/8 (WORKDIR), Step 4/8 (RUN pip install), Step 5/8 (ENV), Step 6/8 (EXPOSE), Step 7/8 (ENTRYPOINT), and Step 8/8 (CMD). The build is successful, and the image is tagged as `manickam19599/bookshelf:latest`.

```
manickam19599@cloudshell:~ (key-tokenizer-344403) $ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
manickam19599/studentserver   latest            2dfab572f804       23 minutes ago     679MB
python               alpine3.7         00be2573e9f7       3 years ago        81.3MB
node                 7                 d9aed20b68a4       4 years ago        660MB
manickam19599@cloudshell:~ (key-tokenizer-344403) $ docker build -t manickam19599/bookshelf .
Sending build context to Docker daemon  169.3MB
Step 1/8 : FROM python:alpine3.7
--> 00be2573e9f7
Step 2/8 : COPY . /app
--> 8f6101b751c0
Step 3/8 : WORKDIR /app
--> Running in ee852cb4408a
Removing intermediate container ee852cb4408a
--> b66a5f6b51fe
Step 4/8 : RUN pip install -r /app/requirements.txt
--> Running in f3931f40d90a
You are using pip version 19.0.1, however version 22.0.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Removing intermediate container f3931f40d90a
--> 9da3884b0c33
Step 5/8 : ENV PORT 5000
--> Running in 37438787419b
Removing intermediate container 37438787419b
--> 27e67Ea35678
Step 6/8 : EXPOSE 5000
--> Running in 2edee30fdd69
Removing intermediate container 2edee30fdd69
--> d7a50e717dd0
Step 7/8 : ENTRYPOINT [ "python3" ]
--> Running in c377306c0008
Removing intermediate container c377306c0008
--> db027292023c
Step 8/8 : CMD [ "bookshelf.py" ]
--> Running in 2239b889ce82
Removing intermediate container 2239b889ce82
--> e49d3b44bd8d
Successfully built e49d3b44bd8d
Successfully tagged manickam19599/bookshelf:latest
manickam19599@cloudshell:~ (key-tokenizer-344403) $
```

```
.docker push manickam19599/bookshelf
```

The screenshot displays a Google Cloud Platform console window with a Cloud Shell terminal. The terminal shows the following steps and output:

```

Step 3/8 : WORKDIR /app
--> Running in ee852cb4408a
Removing intermediate container ee852cb4408a
--> b66a5f6b51fe
Step 4/8 : RUN pip install -r /app/requirements.txt
--> Running in f3931f40d90a
You are using pip version 19.0.1, however version 22.0.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Removing intermediate container f3931f40d90a
--> 9da3884b0c33
Step 5/8 : ENV PORT 5000
--> Running in 37438787419b
Removing intermediate container 37438787419b
--> 27e67fa35678
Step 6/8 : EXPOSE 5000
--> Running in 2edee30fdd69
Removing intermediate container 2edee30fdd69
--> d7a50e717dd0
Step 7/8 : ENTRYPOINT [ "python3" ]
--> Running in c377306c0008
Removing intermediate container c377306c0008
--> db027292023c
Step 8/8 : CMD [ "bookshelf.py" ]
--> Running in 2239b889ce82
Removing intermediate container 2239b889ce82
--> e49d3b44bd8d
Successfully built e49d3b44bd8d
Successfully tagged manickam19599/bookshelf:latest
manickam19599@cloudshell:~ (key-tokenizer-344403) $ docker push manickam19599/bookshelf
Using default tag: latest
The push refers to repository [docker.io/manickam19599/bookshelf]
b6630d562cd9: Pushed
9bbf94e51e20: Pushed
5fa31f02caa8: Mounted from library/python
88e61e328a3c: Mounted from library/python
9b77965e1d3f: Mounted from library/python
50f8b07e9421: Mounted from library/python
629164d914fc: Mounted from library/python
latest: digest: sha256:49ad796e9a24f5c425628ba79685a0929f356fccdcdf047bb95d735862ce0c33 size: 1791
manickam19599@cloudshell:~ (key-tokenizer-344403) $
  
```

The terminal output shows the successful completion of a Docker build and push operation. The image is named 'manickam19599/bookshelf:latest' and is pushed to the Docker Hub repository 'manickam19599/bookshelf'.

studentserver-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  labels:
    app: studentserver-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - image: manickam19599/studentserver
          imagePullPolicy: Always
          name: web
          ports:
            - containerPort: 8080
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_DATABASE
```

Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

studentserver-configmap.yaml

apiVersion: v1

kind: ConfigMap

metadata:

name: studentserver-config

data:

MONGO_URL: <Your EXTERNAL IP ADDRESS
of MONGO-DB Service>

MONGO_DATABASE: mydb

studentserver-service.yaml

apiVersion: v1

kind: Service

metadata:

name: web

spec:

type: LoadBalancer

ports:

service port in cluster

- port: 8080

port to contact inside container

targetPort: 8080

selector:

app: web

bookshelf-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookshelf-deployment
  labels:
    app: bookshelf-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bookshelf-deployment
  template:
    metadata:
      labels:
        app: bookshelf-deployment
    spec:
      containers:
        - image: manickam19599/bookshelf
          imagePullPolicy: Always
          name: bookshelf-deployment
          ports:
            - containerPort: 5000
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: bookshelf-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: bookshelf-config
                  key: MONGO_DATABASE
```

.bookshelf-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: <EXTERNAL IP ADDRESS of MONGODB service>
  MONGO_DATABASE: mydb
```

.bookshelf-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 5000
    # port to contact inside container
    targetPort: 5000
  selector:
    app: bookshelf-deployment
```


Key Deployments to check applications on kubernetes

.mini kube start

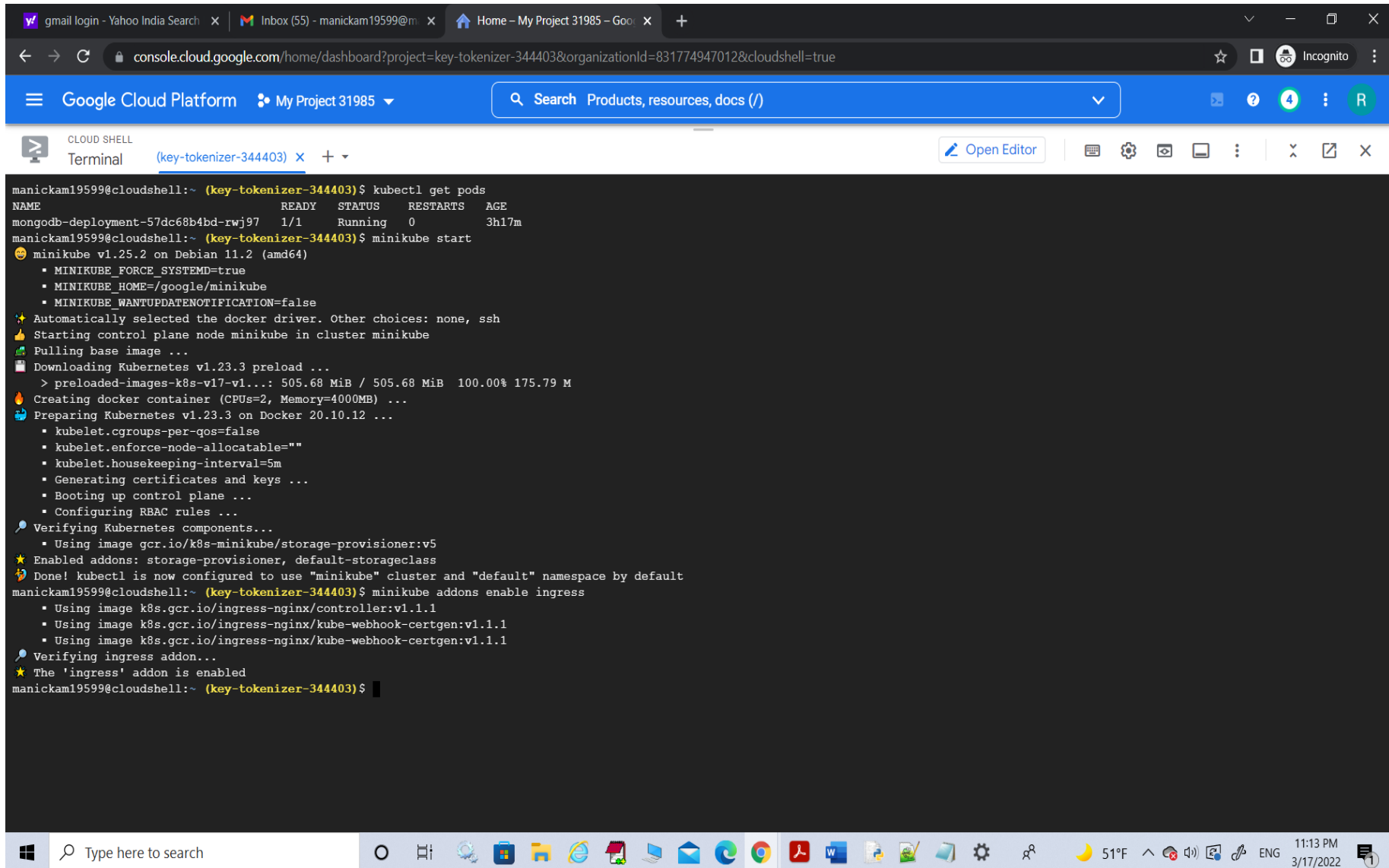
The screenshot shows a web browser window with the Google Cloud Platform console. The address bar shows the URL: `console.cloud.google.com/home/dashboard?project=key-tokenizer-344403&organizationId=831774947012&cloudshell=true`. The page header includes the Google Cloud Platform logo, the project name "My Project 31985", and a search bar. Below the header, there is a "CLOUD SHELL" section with a terminal window titled "Terminal (key-tokenizer-344403)". The terminal output shows the following commands and results:

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-rwj97 1/1     Running   0           3h17m
manickam19599@cloudshell:~ (key-tokenizer-344403)$ minikube start
🐳 minikube v1.25.2 on Debian 11.2 (amd64)
  ▪ MINIKUBE_FORCE_SYSTEMD=true
  ▪ MINIKUBE_HOME=/google/minikube
  ▪ MINIKUBE_WANTUPDATENOTIFICATION=false
🌟 Automatically selected the docker driver. Other choices: none, ssh
👉 Starting control plane node minikube in cluster minikube
📡 Pulling base image ...
📦 Downloading Kubernetes v1.23.3 preload ...
  > preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB 100.00% 175.79 M
🔥 Creating docker container (CPUs=2, Memory=4000MB) ...
🔧 Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  ▪ kubelet.cgroups-per-qos=false
  ▪ kubelet.enforce-node-allocatable=""
  ▪ kubelet.housekeeping-interval=5m
  ▪ Generating certificates and keys ...
  ▪ Booting up control plane ...
  ▪ Configuring RBAC rules ...
🔍 Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
👉 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

The Windows taskbar at the bottom shows the system clock as 11:12 PM on 2/17/2023, with a temperature of 51°F and various system icons.

Start Ingress

minikube addons enable ingress



The screenshot shows the Google Cloud Platform console with a terminal window open. The terminal displays the output of the following commands:

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-rwj97 1/1     Running   0           3h17m
manickam19599@cloudshell:~ (key-tokenizer-344403)$ minikube start
🐳 minikube v1.25.2 on Debian 11.2 (amd64)
  ▪ MINIKUBE_FORCE_SYSTEMD=true
  ▪ MINIKUBE_HOME=/google/minikube
  ▪ MINIKUBE_WANTUPDATENOTIFICATION=false
🌟 Automatically selected the docker driver. Other choices: none, ssh
🔥 Starting control plane node minikube in cluster minikube
📶 Pulling base image ...
📦 Downloading Kubernetes v1.23.3 preload ...
  > preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB 100.00% 175.79 M
🔥 Creating docker container (CPUs=2, Memory=4000MB) ...
🔧 Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  ▪ kubelet.cgroups-per-qos=false
  ▪ kubelet.enforce-node-allocatable=""
  ▪ kubelet.housekeeping-interval=5m
  ▪ Generating certificates and keys ...
  ▪ Booting up control plane ...
  ▪ Configuring RBAC rules ...
🔍 Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
🔥 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
manickam19599@cloudshell:~ (key-tokenizer-344403)$ minikube addons enable ingress
  ▪ Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
  ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
🔍 Verifying ingress addon...
🌟 The 'ingress' addon is enabled
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

The terminal output shows the successful installation and configuration of minikube, including the enabling of the ingress addon. The Google Cloud Platform console interface is visible in the background, showing the project name "key-tokenizer-344403" and the "Terminal" tab selected.

Create studentserver related pods and start service using the above yaml file

kubectl apply -f studentserver-deployment.yaml

kubectl apply -f studentserver-configmap.yaml

kubectl apply -f studentserver-service.yaml

Create bookshelf related pods and start service using the above yaml file

kubectl apply -f bookshelf-deployment.yaml

kubectl apply -f bookshelf-configmap.yaml

kubectl apply -f bookshelf-service.yaml

The screenshot displays the Google Cloud Platform console interface. The top navigation bar shows the project 'My Project 31985'. The left sidebar contains links to 'Kubernetes Engine', 'Marketplace', and 'Release Notes'. The main content area shows the 'Kubernetes Engine' overview for a cluster named 'kubia' in the 'us-west1' region. The cluster has 3 nodes, 6 vCPUs, and 3 GB of memory. A notification indicates 'Can't scale up nodes'. Below the cluster details, a 'CLOUD SHELL' terminal is open, showing a series of kubectl commands and their outputs. The commands are used to apply deployment, configmap, and service manifests for 'studentserver' and 'bookshelf'.

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
✓	kubia	us-west1	3	6	3 GB	⚠ Can't scale up nodes	—

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl apply -f studentserver-service.yaml
service/web created
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
manickam19599@cloudshell:~ (key-tokenizer-344403)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

Check if all the pods are running correctly
kubectrl get pods

The screenshot displays the Google Cloud Platform console interface. At the top, the navigation bar shows 'Google Cloud Platform' and 'My Project 31985'. The left sidebar contains links to 'Kubernetes Engine', 'Marketplace', and 'Release Notes'. The main content area is titled 'Kubernetes Engine' and shows '1 Kubernetes cluster selected'. Below this, there are tabs for 'OVERVIEW' and 'COST OPTIMIZATION'. A filter bar allows searching by property name or value. A table lists the cluster details:

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	kubia	us-west1	3	6	3 GB	Can't scale up nodes	—

Below the table, a 'CLOUD SHELL' terminal window is open, showing the command `kubectl get pods` and its output:

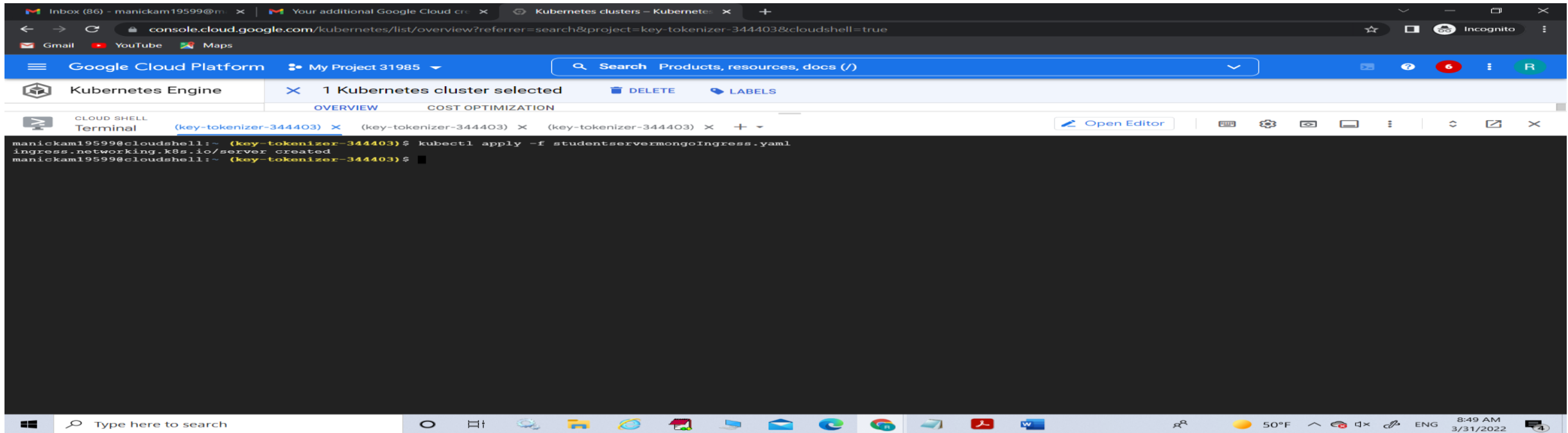
```
manickam19599@cloudshell:~ (key-tokenizer-344403) $ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
bookshelf-deployment-58b84c58fc-dwsn8 1/1     Running   0           14m
mongodb-deployment-57dc68b4bd-tjn9w    1/1     Running   0           71m
web-8699589595-xcwtj                  1/1     Running   0           9m13s
manickam19599@cloudshell:~ (key-tokenizer-344403) $
```

Create an ingress service yaml file called studentservermongoIngress.yaml

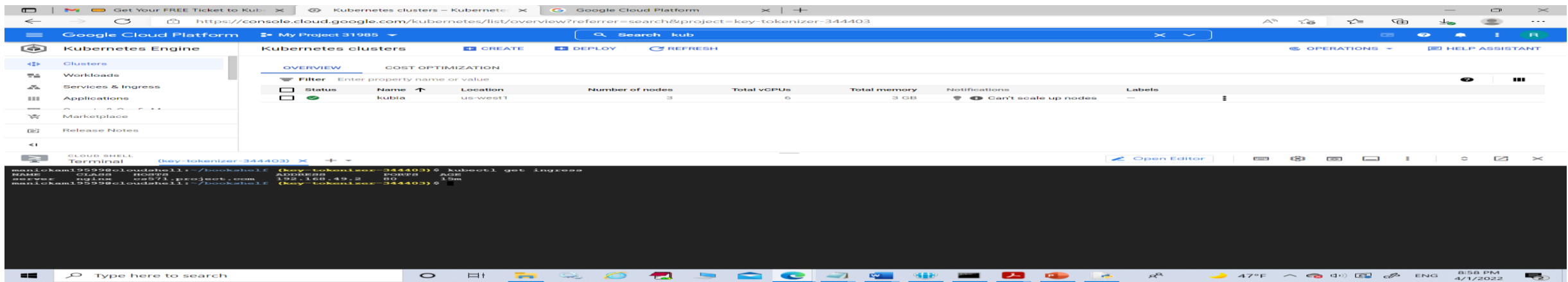
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: cs571.project.com
      http:
        paths:
          - path : /studentserver(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /bookshelf(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000
```

Create the ingress service using the above yamI file

```
kubectl apply -f ../studentservermongoIngress.yamI
```



Check if ingress is running
kubectrl get ingress
Please wait until you see the Address, then move forward



Add Addressee to /etc/hosts

vi /etc/hosts

Add the address you got from above step to the end of the file

Your-address cs571.project.com

Your /etc/hosts file should look something like this after adding the line, but your address should be different from mine

The screenshot shows the Google Cloud Platform console for project 'key-tokenizer-344403'. The 'Kubernetes Engine' section is active, displaying a table of clusters. One cluster named 'kubia' is shown with 3 nodes and 6 vCPUs. Below the table, a terminal window is open, showing the output of 'kubectl get ingress' and the contents of the '/etc/hosts' file. The terminal output includes the following lines:

```
manickam19599@cloudshell:~/bookshelf (key-tokenizer-344403)$ kubectl get ingress
NAME      CLASS    HOSTS                ADDRESS      PORTS      AGE
server    nginx    cs571.project.com    192.168.49.2  80         6m47s
manickam19599@cloudshell:~/bookshelf (key-tokenizer-344403)$ cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1    localhost
::1         localhost ip6-localhost ip6-loopback
fe00::0     ip6-localnet
fe00::0     ip6-mcastprefix
fe00::1     ip6-allnodes
fe00::2     ip6-allrouters
172.17.0.4   cs-610802588758-default
192.168.49.2 cs571.project.com
manickam19599@cloudshell:~/bookshelf (key-tokenizer-344403)$
```

The bottom of the image shows the Windows taskbar with the search bar and various application icons.

If everything goes smoothly, you should be able to access your applications

```
curl cs571.project.com/studentserver/api/score?student_id=11111
```

```
curl cs571.project.com/studentserver/api/score?student_id=22222
```

```
curl cs571.project.com/studentserver/api/score?student_id=33333
```

The screenshot shows the Google Cloud Platform console interface. At the top, there are tabs for 'Get Your FREE Ticket to Kube...', 'Kubernetes clusters - Kubernetes...', and 'Google Cloud Platform'. The main navigation bar includes 'Google Cloud Platform', 'My Project 31985', and a search bar containing 'kub'. Below the navigation bar, there is a 'CLOUD SHELL Terminal' section for the project 'key-tokenizer-344403'. The terminal window displays the following commands and outputs:

```
manickam19599@cloudshell:~ (key-tokenizer-344403) $ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"6247cf7febb186a5d0054b2d","student_id":11111,"student_name":"Bruce Lee","grade":84}
manickam19599@cloudshell:~ (key-tokenizer-344403) $ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"6247cf7febb186a5d0054b2e","student_id":22222,"student_name":"Jackie Chen","grade":93}
manickam19599@cloudshell:~ (key-tokenizer-344403) $ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"6247cf7febb186a5d0054b2f","student_id":33333,"student_name":"Jet Li","grade":88}
manickam19599@cloudshell:~ (key-tokenizer-344403) $
```

The bottom of the image shows a Windows taskbar with the search bar 'Type here to search' and various application icons. The system clock in the bottom right corner indicates the time is 9:40 PM on 4/1/2022, with a temperature of 47°F.

On another path, you should be able to use the REST API with bookshelf application
I.e list all books
`curl cs571.project.com/bookshelf/books`

The screenshot displays the Google Cloud Platform console interface. The top navigation bar includes the Google Cloud Platform logo, the project name 'My Project 31985', a search bar with the text 'kub', and various utility icons. The left sidebar contains navigation links for 'Kubernetes Engine', 'Marketplace', and 'Release Notes'. The main content area is titled 'Kubernetes clusters' and features tabs for 'OVERVIEW' and 'COST OPTIMIZATION'. A table lists the clusters, with one cluster named 'kubia' in the 'us-west1' region, having 3 nodes, 6 vCPUs, and 3 GB of memory. A notification indicates 'Can't scale up nodes'. Below the table, a 'CLOUD SHELL' terminal window is open, showing the command `curl cs571.project.com/bookshelf/books` and its output, which is a JSON array containing one book object.

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	kubia	us-west1	3	6	3 GB	Can't scale up nodes	-

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6247caf1ebb6d7f9bd1edee4"
  }
]
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

Add a book

curl -X POST -d '{"book_name\": \"cloud computing\", \"book_author\": \"unkown\", \"isbn\": \"123456\" }' http://cs571.project.com/bookshelf/book

Update a

Google Cloud Platform

My Project 31985

Search kub

Kubernetes Engine

Marketplace

Release Notes

Kubernetes clusters

CREATE

DEPLOY

REFRESH

OPERATIONS

HELP ASSISTANT

OVERVIEW

COST OPTIMIZATION

Filter

Enter property name or value

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	kubia	us-west1	3	6	3 GB	Can't scale up nodes	

CLOUD SHELL

Terminal

(key-tokenizer-344403)

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl -X POST -d '{"book_name\": \"cloud computing\", \"book_author\": \"unkown\", \"isbn\": \"123456\" }' http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

Update a book

curl -X PUT -d '{"book_name\": \"123\", \"book_author\": \"test\", \"isbn\": \"123updated\" }' http://cs571.project.com/bookshelf/book/id

Google Cloud Platform

My Project 31985

Search kub

Kubernetes Engine

Marketplace

Release Notes

Kubernetes clusters

CREATE

DEPLOY

REFRESH

OVERVIEW

COST OPTIMIZATION

Filter

Enter property name or value

Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	kubia	us-west1	3	6	3 GB	Can't scale up nodes	

CLOUD SHELL

Terminal

(key-tokenizer-344403)

Open Editor

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6247caf1ebb6d7f9bd1edee4"
  }
]
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl -X PUT -d '{"book_name\": \"123\", \"book_author\": \"test\", \"isbn\": \"123updated\" }' http://cs571.project.com/bookshelf/book/6247caf1ebb6d7f9bd1edee4
{
  "message": "Task updated successfully!"
}
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl cs571.project.com/bookshelf/6247caf1ebb6d7f9bd1edee4
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "6247caf1ebb6d7f9bd1edee4"
  }
]
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

Delete a book

```
curl -X DELETE cs571.project.com/bookshelf/book/id
```

The screenshot shows the Google Cloud Platform console interface. At the top, there are tabs for 'Inbox (87) - manickam19599@m...' and 'Kubernetes clusters - Kubernetes'. The main header bar displays 'Google Cloud Platform' and 'My Project 31985'. A search bar contains the text 'Search kub'. Below the header, the 'Kubernetes Engine' section is active, showing '1 Kubernetes cluster selected' with 'DELETE' and 'LABELS' buttons. A 'CLOUD SHELL' terminal window is open, displaying the following commands and output:

```
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "62468f2e0b866463e50dd707"
  }
]
manickam19599@cloudshell:~ (key-tokenizer-344403)$ curl -X DELETE cs571.project.com/bookshelf/book/62468f2e0b866463e50dd707
{
  "message": "Task deleted successfully!"
}
manickam19599@cloudshell:~ (key-tokenizer-344403)$
```

Reference

SFBU Course Material